Analysis of Coppersmith's Block Wiedemann Algorithm for the Parallel Solution of Sparse Linear Systems

Erich Kaltofen

Rensselaer Polytechnic Institute
Department of Computer Science
Troy, New York, USA

Outline

- **object representation**
  - black box representations for sparse matrices
- **Wiedemann's algorithm**
  - homogeneous linear systems
- **Coppersmith's blocking**
  - my probabilistic analysis
  - fast solution of singular block-Toeplitz systems
- **our implementation efforts in DSC**
  - timings for test cases (with A. Lobo [DISCO '93])
- **the large sparse linear system challenge**

What is a sparse matrix?

- **matrices with "few" non-zero entries**
  - a band matrix from a finite element method
  - a matrix over GF(2) from integer factoring by the NFS: $52\,250 \times 50\,001$ with $1\,095\,532$ entries $\neq 0$ (≈ 21/row)

- **matrices with special structure**
  - the Sylvester matrix corresponding to a polynomial resultant

$$R = \begin{bmatrix} a_N & a_{N-1} & \cdots\cdots & a_0 & & & \\ & a_N & \cdots\cdots & a_1 & a_0 & & \mathbf{0} \\ & \mathbf{0} & \ddots & & \ddots & \ddots & \\ & & & a_N & \cdots\cdots\cdots & & a_0 \\ b_N & b_{N-1} & \cdots\cdots & b_0 & & & \\ & b_N & \cdots\cdots & b_1 & b_0 & & \mathbf{0} \\ & \mathbf{0} & \ddots & & \ddots & \ddots & \\ & & & b_N & \cdots\cdots\cdots & & b_0 \end{bmatrix}$$

- **a “black box” matrix**
  an efficient program with the specifications

$$B \in \mathsf{K}^{N \times N}$$

K an arbitrary field

e.g., for the Sylvester matrix R, $R \cdot b$ costs

$$O(N \, \log N \, \operatorname{loglog} N)$$

arithmetic operations using fast polynomial multiplication

Symbolic objects given by black box representation are known for many problems:

- symbolic determinants using Gaussian elimination

- the polynomial remainder sequence of $f_0(x)$ and $f_1(x)$ using continued fraction approximations

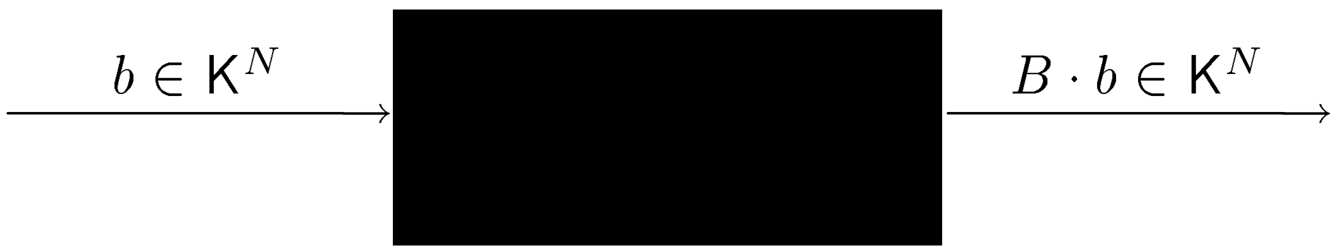
$$\{q_i(x)\}_{i\geq 2} \quad \text{such that} \quad f_i(x) = f_{i-2}(x) - q_i(x) f_{i-1}(x)$$

- $B^{-1} = P^{-1}U^{-1}L^{-1}$, the LUP factorization of $B \in \mathsf{K}^{N\times N}$.

- streams for infinite objects, such as a program for the i-th order coefficient of a power series

Linear system solution with a black box matrix

Given a black box



$B \in \mathsf{K}^{N \times N}$ singular
K an arbitrary field

compute $w \neq \mathbf{0}$ such that $Bw = \mathbf{0}$ "efficiently."

D. Wiedemann (1986) constructs a Las-Vegas-randomized algorithm that computes w in at most

$$3N \text{ "}B \cdot b \text{ steps"}$$

and

$$O(N^2) \text{ additional arithmetic operations in } \mathsf{K}.$$

The algorithm needs $O(N)$ space.

Idea for WIEDEMANN's algorithm

$B \in \mathsf{K}^{N\times N}$, K a finite field

$f^B(\lambda) = c'_0 + c'_1\lambda + \cdots + c_{M'}\lambda^{M'} \in \mathsf{K}[\lambda]$ **minimum polynomial** of B:

$$\forall\, u, v \in \mathsf{K}^N\colon\ \forall\, j \geq 0\colon\ u^{\mathrm{tr}} B^j f^B(B) v = 0$$

$$\Updownarrow$$

$$c'_0\, \underbrace{u^{\mathrm{tr}} B^j v}_{a_j} + c'_1\, \underbrace{u^{\mathrm{tr}} B^{j+1} v}_{a_{j+1}} + \cdots + c'_{M'}\, \underbrace{u^{\mathrm{tr}} B^{j+M'} v}_{a_{j+M'}} = 0$$

$$\Updownarrow$$

$\{a_0, a_1, a_2, \ldots\}$ is generated by a linear recursion

Theorem [Wiedemann 1986]: *For random* $u, v \in \mathsf{K}^N$, *a linear generator for* $\{a_1, a_1, a_2, \ldots\}$ *is one for* $\{I, B, B^2, \ldots\}$.

$$\forall\, j \geq 0\colon c_0 a_j + c_1 a_{j+1} + \cdots + c_M a_{j+M} = 0$$

$$\Big\Downarrow \text{ (with high probability)}$$

$$c_0 B^j v + c_1 B^{j+1} v + \cdots + c_M B^{j+M} v = \mathbf{0}$$

$$\Big\Downarrow \text{ (with high probability)}$$

$$c_0 B^j + c_1 B^{j+1} + \cdots + c_M B^{j+M} = \mathbf{0}$$

that is, $f^B(\lambda)$ divides $c_0 + c_1\lambda + \cdots + c_M\lambda^M$

One may compute the c_j from the a_j by the Berlekamp/Massey algorithm, or by solving a homogenous linear Toeplitz system:

$$\begin{bmatrix} a_M & \dots & & a_1 & a_0 \\ a_{M+1} & a_M & & a_2 & a_1 \\ \vdots & & \ddots & & \vdots \\ a_{2M-1} & \dots & & & a_{M-1} \end{bmatrix} \underbrace{\begin{bmatrix} c_M \\ c_{M-1} \\ \vdots \\ c_0 \end{bmatrix}}_{\neq 0} = \mathbf{0}$$

for any $M \geq \deg(f^B)$, in particular for $M = N$.

Algorithm *Homogeneous Wiedemann*

Input: $B \in \mathsf{K}^{N \times N}$ singular

Output: $w \neq \mathbf{0}$ such that $Bw = \mathbf{0}$

Step W1: Pick random $u, v \in \mathsf{K}^N$; $\quad b \leftarrow Bv$;
for $i \leftarrow 0$ **to** $2N-1$ **do** $a_i \leftarrow u^{\mathrm{tr}} B^i b$.

$2N$ "$B \cdot y$" steps
$O(N^2)$ arithm. op's

Step W2: Compute a linear recurrence generator for $\{a_i\}$,
$c_L \lambda^L + c_{L+1} \lambda^{L+1} + \cdots + c_D \lambda^D, \quad L \geq 0, c_L \neq 0.$

$O(N (\log N)^2 \operatorname{loglog} N)$ arithm. op's

Step W3: $\widehat{w} \leftarrow c_L v + c_{L+1} Bv + \cdots + c_D B^{D-L} v$;
(With high probability $\widehat{w} \neq 0$ and $B^{L+1} \widehat{w} = 0$)
Compute first l with $B^l \widehat{w} = 0$; **return** $w \leftarrow B^{l-1} \widehat{w}$.

$\leq N+1$ "$B \cdot y$" steps
$O(N^2)$ arithm. op's

Other applications of "coordinate recurrences"

- solution of sparse singular inhomogeneous linear systems [K & Saunders 1991]
- processor-efficient parallel poly-log-time algorithms for dense linear systems [K & Pan 1991, 1992]
- Frobenius form computation of an $N \times N$ matrix in $O(N^{2.375})$ field operations [Giesbrecht 1991]
- processor-efficient parallel poly-log-time algorithms for the characteristic polynomial [Eberly 1991, Giesbrecht 1992]
- space-complexity improvement of the Berlekamp polynomial factoring algorithm [K 1991]

Coppersmith's (1992) parallelization (modified)

Use of the block vectors $\boldsymbol{x} \in \mathsf{K}^{N\times m}$ in place of u
$\boldsymbol{z} \in \mathsf{K}^{N\times n}$ in place of v

$$\boldsymbol{a}_i = \boldsymbol{x}^{\mathrm{tr}} B^{i+1} \boldsymbol{z} \in \mathsf{K}^{m\times n}$$

Find a vector polynomial $c_L\lambda^L + c_{L+1}\lambda^{L+1} + \cdots + c_D\lambda^D \in \mathsf{K}^n[\lambda]$, such that

$$\forall\, j \geq 0\colon \sum_{i=L}^{D} \boldsymbol{a}_{j+i} c_i = \sum_{i=L}^{D} \boldsymbol{x}^{\mathrm{tr}} B^{i+j}\, B\boldsymbol{z} c_i = \mathbf{0} \in \mathsf{K}^{m\times n}$$

Then, analogously to before, with high probability

$$\widehat{w} = \sum_{i=L}^{D} B^{i-L}\, \boldsymbol{z} c_i \neq \mathbf{0}, \quad B^{L+1}\widehat{w} = \sum_{i=L}^{D} B^i B \boldsymbol{z} c_i = \mathbf{0} \quad \in \mathsf{K}^N$$

Questions

- how many $\boldsymbol{a}_i$ are needed?
- how can the c_i be computed from the $\boldsymbol{a}_i$?
- probability of success? does method work on "pathological" B's?

- iteration length: $i \le \dfrac{N}{m} + \dfrac{N}{n} + \dfrac{2n}{m} + 1$

- computation of c_i:
  - Coppersmith has generalized the Berlekamp/Massey algorithm
  - Block-Toeplitz solver by Kailath et al. (1979)
    Also yields the rank of B.

  Both methods require $O((m+n)N^2)$ arithmetic operations in K or with $m+n$ processors $O(N^2)$ parallel time.

  - Divide-and-conquer Toeplitz-like solver [Bitmead & Anderson 1980, Morf 1980, K. 1993]:

  Requires $O((m+n)^2 N (\log N)^2 \operatorname{loglog} N)$ arithmetic ops. in K.

Probabilistic analysis

Theorem: *If B is singular with*

$$\deg f^B = 1 + \operatorname{rank} B \tag{1}$$

then we find $w \neq \mathbf{0}$ with $Bw = \mathbf{0}$ for random $\boldsymbol{x}$, $\boldsymbol{z}$ with probability

$$\geq 1 - \frac{1 + 2 \operatorname{rank} B}{\operatorname{card} \mathsf{K}} \geq 1 - \frac{2N - 1}{\operatorname{card} \mathsf{K}}$$

Condition (1) can be enforced by "randomly mixing" B à la Beneš/ Wiedemann (1986), for instance

$\widetilde{B} \leftarrow V \cdot B \cdot W \cdot G$ where V realizes random row permut. network
W realizes random col. permut. network
G is random diagonal

Parallel coarse-grain realization

The ν^{th} processor computes the ν^{th} column of $\boldsymbol{a}_i$, $i \lessapprox \dfrac{N}{m} + \dfrac{N}{n}$

Running-time comparisions

	Wiedemann		seq. blocked W. m, n fixed $\varepsilon = \frac{n}{m} + \frac{1}{n} < 1$	par. blocked W. n processors $\frac{m}{n}$ fixed
# of $B \cdot y$ products	$2N$	$3N$	$(1+\varepsilon)N + 2$	$\frac{N}{m} + \frac{N}{n} + O(1)$
# of arithm. operations	$O(N^2)$	$O(N^2)$	$O_\varepsilon(N^2 \log N)$	$O((1 + \frac{\log N}{n})N^2)$ (parallel w/o FFT) $O(n^2 N(\log N)^2 \times \operatorname{loglog} N)$ (sequent. w. FFT)
amount of storage	$O(N^2)$	$O(N)$	$O_\varepsilon(N)$	$O(nN)$

Proof idea for probabilistic analysis

$\sum_{i=0}^{N/n} B^{i+1} \boldsymbol{z} c_i = \mathbf{0}$ is a linear condition on $c_i \equiv$ block-Krylov system

If rank(block-Toeplitz) = rank(block-Krylov)
then every solution of block-Toeplitz system is one for block-Krylov system

The *generic* block-Toeplitz/Krylov systems can be specialized to the *generic* Toeplitz system of the Wiedemann algorithm

Thus the rank condition holds for the generic systems, hence for the ones obtained at random specializations by the Schwartz/Zippel lemma.

DSC features

- uses low level UNIX IP/TCP/UDP process communication
- can distribute C and Lisp source code
- network can be interactively monitored for progress and faults
- computers are auto-selected w.r.t. local resources and work load
  implemented by A. Diaz and M. Hitz [DISCO '93]
- supports co-routine calling mechanism
  implemented by A. Diaz [DISCO '93]
- messages are digitally signed for secure communication

Test 1: sparse random matrices over GF(32 749)

N	Task	Blocking Factor 2	4	8
$10\,000^{\dagger}$	(1) $\langle \boldsymbol{a}_i \rangle$	7:29	3:54	2:09
	(2) b-massey	2:25	4:08	8:00
	(3) evaluation	3:47	1:59	1:05
	total	13:41	10:06	11:14
$20\,000^{\ddagger}$	(1) $\langle \boldsymbol{a}_i \rangle$	57:17	28:43	15:21
	(2) b-massey	9:48	16:36	33:39
	(3) evaluation	29:42	14:44	7:53
	total	96:47	60:02	56:53

CPU Time for different blocking factors in *hours:minutes*
each processor rated at 28.5 MIPS
$\dagger \approx 350\,000$ non-zero entries
$\ddagger \approx 1\,300\,000$ non-zero entries

Test 2: sparse matrices over GF(2)

N	Task	Blocking Factor 1×32	2×32	3×32
20 000‡	(1) $\langle \boldsymbol{a}_i \rangle$	1:12	0:40	0:30
	(2) b-massey	0:25	0:31	0:39
	(3) evaluation	0:29	0:28◇	0:10
	total	2:06	1:39	1:19
52 250*	(1) $\langle \boldsymbol{a}_i \rangle$	3:53	2:11	1:37
	(2) b-massey	2:30	3:09	3:54
	(3) evaluation	1:15	0:33	0:22
	total	7:38	5:53	5:53

CPU Time for different blocking factors in *hours:minutes*
32 bit operations performed simultan. as one computer word op.
‡ $\approx$ 1 300 000 non-zero entries picked at random
$*$ from NFS integer factoring; $\approx$ 1 100 000 non-zero entries
note: unblocked Wiedemann algorithm takes $\approx$ 111 hours
◇ first $\widehat{w} = \mathbf{0}$.

Test 3: very large sparse matrix over GF(2)

N	Task	Blocking Factor 1×32	2×32	3×32
$100,000^\star$	(1) $\langle a^{(i)} \rangle$	77:37	44:05	27:28
	(2) b-massey	10:03	12:28	15:42
	(3) evaluation	74:37	27:48	11:09
	total	162:17	84:31	54:19

CPU Time for different blocking factors in *hours:minutes*
each processor rated at 28.5 MIPS
32 bit operations performed simultan. as one computer word op.
$\star$ 10 304 243 non-zero entries picked at random

The large sparse linear system challenge

Solve a sparse $100\,000 \times 100\,000$ linear system over $GF(2^{32}-5)$ with $10\,000\,000$ non-zero entries.

Note Amdahl's law:

$$\frac{T_{\text{par}}}{T_{\text{seq}}} = \alpha + \frac{1-\alpha}{p(1-c)} \quad \text{where} \quad \begin{array}{ll} p & \text{\# of processors} \\ \alpha & \text{ratio spent in seq. part} \\ c & \text{ratio of par. part spent in commun.} \end{array}$$